## Data Analysis with Python

### Cheat Sheet: Model Development

| Process | Description | Code Example |
|---|---|---|
| Linear Regression | Create a Linear Regression model object | `from sklearn.linear_model import LinearRegression`<br>`lr = LinearRegression()` |
| Train Linear Regression model | Train the Linear Regression model on decided data, separating Input and Output attributes. When there is single attribute in input, then it is simple linear regression. When there are multiple attributes, it is multiple linear regression. | `X = df[['attribute_1', 'attribute_2', ...]]`<br>`Y = df['target_attribute']`<br>`lr.fit(X,Y)` |
| Generate output predictions | Predict the output for a set of Input attribute values. | `Y_hat = lr.predict(X)` |
| Identify the coefficient and intercept | Identify the slope coefficient and intercept values of the linear regression model defined by $\hat{y} = mx + c$ Where m is the slope coefficient and c is the intercept. | `coeff = lr.coef`<br>`intercept = lr.intercept_` |
| Residual Plot | This function will regress y on x (possibly as a robust or polynomial regression) and then draw a scatterplot of the residuals. | `import seaborn as sns`<br>`sns.residplot(x=df[['attribute_1']],`<br>`y=df[['attribute_2']])` |
| Distribution Plot | This function can be used to plot the distribution of data w.r.t. a given attribute. | `import seaborn as sns`<br>`sns.distplot(df['attribute_name'], hist=False)`<br>`# can include other parameters like color, label and so on.` |
| Polynomial Regression | Available under the numpy package, for single variable feature creation and model fitting. | `f = np.polyfit(x, y, n)`<br>`#creates the polynomial features of order n`<br>`p = np.poly1d(f)`<br>`#p becomes the polynomial model used to generate the predicted output`<br>`Y_hat = p(x)`<br>`# Y_hat is the predicted output` |
| Multi-variate Polynomial Regression | Generate a new feature matrix consisting of all polynomial combinations of the features with the degree less than or equal to the specified degree. | `from sklearn.preprocessing import PolynomialFeatures`<br>`Z = df[['attribute_1','attribute_2',...]]`<br>`pr=PolynomialFeatures(degree=n)`<br>`Z_pr=pr.fit_transform(Z)` |
| Pipeline | Data Pipelines simplify the steps of processing the data. We create the pipeline by creating a list of tuples including the name of the model or estimator and its corresponding constructor. | `from sklearn.pipeline import Pipeline`<br>`from sklearn.preprocessing import StandardScaler`<br>`Input=[('scale',StandardScaler()), ('polynomial',`<br>`PolynomialFeatures(include_bias=False)),`<br>`('model',LinearRegression())]`<br>`pipe=Pipeline(Input)`<br>`Z = Z.astype(float)`<br>`pipe.fit(Z,y)`<br>`ypipe=pipe.predict(Z)` |
| R^2 value | R^2, also known as the coefficient of determination, is a measure to indicate how close the data is to the fitted regression line.<br>The value of the R-squared is the percentage of variation of the response variable (y) that is explained by a linear model.<br>a. For Linear Regression (single or multi attribute)<br>b. For Polynomial regression (single or multi attribute) | a.<br>`X = df[['attribute_1', 'attribute_2', ...]]`<br>`Y = df['target_attribute']`<br>`lr.fit(X,Y)`<br>`R2_score = lr.score(X,Y)`<br><br>b.<br>`from sklearn.metrics import r2_score`<br>`f = np.polyfit(x, y, n)`<br>`p = np.poly1d(f)`<br>`R2_score = r2_score(y, p(x))` |
| MSE value | The Mean Squared Error measures the average of the squares of errors, that is, the difference between actual value and the estimated value. | `from sklearn.metrics import mean_squared_error`<br>`mse = mean_squared_error(Y, Yhat)` |

Skills Network

IBM